1. Introduction:

The automation framework for Monday.com is designed to automate the testing of a web application that manages budgets and associated actions. The framework is implemented using the Page Object Model (POM) design pattern, ensuring a modular and maintainable structure. It utilizes XPath locators based on text for improved reusability and generality across the application.

2. Folder Structure:

The framework follows a well-organized folder structure to enhance code readability, ease of maintenance, and separation of concerns. The main folders are as follows:

src/main/java: Contains the main Java code for base class, page objects, utilities, and configuration files.

com.main.pages.loginPage: Page class to keep all the methods related to login page

com.main.pages.sampleBoardPage: Page class to keep all the methods related to sample board page

com.main.configurations.configuration: Package for storing configuration files (e.g., properties files).

src/test/java: Holds the test cases in separate classes based on functionality.

allure-results: The directory to store Allure report results.

allure-report: The directory containing the generated Allure HTML report.

1. Page Object Model (POM):

The POM design pattern is implemented in this framework to ensure maintainability and reusability. Each web page is represented by a separate Java class in the pages package. The class encapsulates the page elements and actions as methods. The test classes in the tests package interact with these page objects to perform actions and validations.

1. Methods :

The framework leverages the use of generic methods to enhance code reusability and minimize duplication. Some examples of generic methods include:

addGroupName: A method that add Budget name against any budget Group

setStatusOfBudget: Method to set status of any to budget

selectRandomDateFromDatePicker: A method that selects a random date from the visible dates in a date picker on the page.

addFormula: A method that adds a formula to the specified budget by interacting with the CodeMirror editor.

1. Allure Reporting:

The framework utilizes Allure reporting to generate comprehensive and visually appealing reports for test results. Allure annotations, such as @Step, @Feature, @Story, and @Description, are used to provide detailed context and descriptions for each test step. The allure-report folder contains the generated HTML reports, including screenshots and videos for test execution.

1. TestNG Test Execution:

TestNG is used as the test runner to execute the test cases defined in the test classes. The testNg.xml file is used to configure the test suite, including defining test classes, specifying parallel execution, and setting up listeners for Allure reporting.

Open the testng.xml file:

Navigate to the project's root directory where the testng.xml file is located.

Open the testng.xml file using a text editor or an IDE.

Configure Test Suite (Optional):

Within the testng.xml file, you can configure the test suite as per your requirement.

You can specify the desired test classes to be executed and set any other TestNG parameters, such as parallel execution or data providers.

Run the Test Suite

Save the changes made to the testng.xml file.

TestNG will start running the test cases based on the configurations specified in the testng.xml file.

Monitor Test Execution:

As the tests run, you will see the test execution progress and status displayed in the terminal.

TestNG will execute the test methods defined in the test classes and generate the Allure report.

View Allure Report:

After the test execution is complete, you can find the generated Allure report in the allure-report directory.

Open the index.html file from the allure-report directory in a web browser to view the detailed Allure report with test results, screenshots, and videos.